Algorithme et Langage C	Les structures de contrôle ~ Schéma conditionnel	Fiche n° 1

Notions de prédicat :

*Un prédicat est l'expression d'une condition. Il peut prendre l'une des deux valeurs **VRAIE** ou **FAUSSE**. Les conditions sont exprimées sous la forme d'une comparaison de deux termes. Si nécessaire, lorsqu'il y a plus de deux termes à comparer, le prédicat devient un prédicat composé à l'aide des connecteurs **ET** et **OU**. Pour exprimer une condition inverse, le connecteur **NON** permet d'inverser cette condition.*

Comparaisons	Algorithmique	Langage C
Égale	=	==
Différent	≠	!=
Inférieur	<	<
Inférieur ou égale	≤	<=
Supérieur	>	>
Supérieur ou égale	≥	>=

Connecteurs	Langage C
ET	&&
OU	||
NON	!

Il est parfois nécessaire d'utiliser plusieurs niveaux de parenthèse pour respecter la priorité entre les connecteurs ***ET*** et les ***connecteurs OU***.
Exemple : *(val1 = 0 ou val2 = 0) et val3 > 4*

Schéma conditionnel : Si ... Alors ...

Ce schéma conditionnel permet de prendre une décision suite à l'évaluation d'un prédicat. Il est nécessaire de pratiquer une indentation pour faciliter la lecture. Le traitement peut être composé de plusieurs instructions (les unes en dessous des autres en respectant l'indentation), il se termine au FinSi en algorithme.

Si l'évaluation du prédicat est vraie alors le traitement est réalisé

```
Si prédicat
   Alors traitement
FinSi
```

```
if ( /* prédicat*/    )
{
    /* traitement */
}
```

Le prédicat est toujours exprimé entre parenthèses en langage C. Il n'y a pas de point-virgule à la suite. Le **Alors** n'est pas traduit explicitement en langage C

Exemples :

val1 $_{entier}$
val2 $_{entier}$

```
Si val1 = 5
   Alors   val1 ← 0
           val2 ← 2
FinSi
```

```
if (val1 == 5)
{
    Val1 = 0 ;
    Val2 = 2 ;
}
```

Le bloc d'instructions est défini entre accolades, il représente le traitement à effectuer lorsque le prédicat est VRAI.

Si le traitement est composé d'une seule instruction, les accolades ne sont pas obligatoires.

```
Si  val1 = 5 ET val2 ≠  0
   Alors val1 = val1 + 1
FinSi
```

```
if  (val1 == 5 && val2 != 0)
    Val1++ ;
```

Les mots clés du langage C sont toujours écrits en minuscule.
Chaque instruction en langage C se termine par un point-virgule.

Schéma conditionnel : Si … Alors … Sinon …

Ce schéma conditionnel permet une alternative entre deux traitements en fonction de l'évaluation d'un prédicat. Si la condition représentant le prédicat est vraie, le premier traitement est réalisé sinon c'est le second. Il est possible d'imbriquer plusieurs schémas conditionnels les uns dans les autres.

Le traitement 1 est réalisé jusqu'au **Sinon** si la valeur du prédicat est vraie.

Si elle est fausse, le traitement 2 est effectué jusqu'au **FinSi.**

```
Si prédicat
   Alors traitement 1
   Sinon traitement 2
FinSi
```

```
if (  /* prédicat*/   )
{
    /* traitement 1 */
}
else
{
   /* traitement 2 */
}
```

Deux blocs distincts entre accolades différencient les traitements.

Seul un traitement composé de plusieurs instructions impose des accolades.

L'indentation est nécessaire pour la lisibilité du code.

Imbrications

```
Si prédicat 1
   Alors traitement 1
   Sinon Si prédicat 2
            Alors traitement 2
         FinSi
FinSi
```

```
if (  /* prédicat 1 */  )
{
    /* traitement 1 */
}
else{ if ( /* prédicat 2 */ )
      {
         /* traitement 2 */
      }
}
```

Dans ce cas, le traitement 2 est conditionné à l'évaluation d'un deuxième prédicat.
Le traitement 2 ne peut être réalisé que si le prédicat 1 est faux.
Il n'y a qu'une instruction dans le ***else*** les accolades ne sont donc pas obligatoires.

Schéma conditionnel généralisé : cas … parmi …

C'est la généralisation de l'imbrication des schémas conditionnels présentés précédemment. Ce schéma est cependant limité à l'évaluation d'une valeur constante. Les traitements qui en suivent sont conditionnés par la valeur de cette constante. Un traitement par défaut peut être envisagé si aucune des valeurs listées ne correspond à la valeur prise par la variable, cette rubrique est facultative cependant.

```
Cas nomDeVariable parmi
    VAL1 : traitement 1
    VAL2 : traitement 2
    VAL3 : traitement 3
    …
    Par défaut : traitement par défaut
FinCas
```

VAL1, VAL2, VAL3 représentent ici des valeurs constantes que peut prendre la variable désignée par *nomDeVariable*.

Ces constantes sont limitées aux entiers et aux simples caractères.

Si *nomDeVariable* est égale à VAL1 c'est le traitement 1 qui est réalisé sinon si elle est égale à VAL2 c'est le 2 et ainsi de suite. Si aucun des cas ne correspond, c'est le traitement par défaut.

En algorithmique, les traitements s'arrêtent à l'énoncé du cas suivant. L'instruction cas… parmi est dite ici exclusive.

```
switch (nomDeVariable)
{
    case VAL1 :    /* traitement 1 */
                   break;
    case VAL2 :    /* traitement 2 */
                   break;
    case VAL3 :    /* traitement 3 */
                   break;
    /* … */
    default :      / * traitement  par défaut */
}
```

L'instruction ***break*** est nécessaire pour ne pas enchaîner les traitements. En effet, le ***switch*** … ***case*** en langage C est dit inclusif. Les traitements ne sont pas marqués sous forme de bloc entre accolades et ne s'arrêtent pas au ***case*** suivant. Le ***break*** est donc nécessaire pour stopper la suite d'instructions.
Le ***default*** est facultatif, si aucun cas n'est possible, il est exécuté.